NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

REASONING BY ANALOGY USING HOLOGRAPHIC CONCEPTUAL PROJECTION

by

Yilmaz Degirmenci

September 2002

Thesis Advisor:	Neil Rowe
Second Reader:	John Hiles

Approved for public release; distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** September 2002	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE: Reasoning by Analogy Using Holographic Conceptual Projection			**5. FUNDING NUMBERS**
6. AUTHOR(S)			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited			**12b. DISTRIBUTION CODE**

13. ABSTRACT

This thesis discusses the designing of an architecture which mimics a human thought mechanism. The architecture is called a Holographic Conceptual Projection, which uses analogy and dynamic pattern matching combined with some natural-language understanding. Our main hypothesis is that we project our way of thinking into words and sentences which we manipulate when thinking verbally. This means we can exploit the structure of sentences to build an algorithm that models our thought mechanism. In our Holographic Conceptual Projection Architecture we give examples of every word within the context patterns. The patterns contain sentences that describe the "condition", "desired situation", "proposition" and "outcome" of the concept. The concept's patterns are then compared with new cases to see analogies. This comparison is done with dynamic generalization and specialization techniques. Finally after building an implementation, we tested it on an intelligent file-management system and an image-processing application.

14. SUBJECT TERMS Artificial Intelligence, Reasoning by Analogy, Conceptual Projection, Means-Ends Analysis, Linguistics.			**15. NUMBER OF PAGES** 60
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited.

REASONING BY ANALOGY USING HOLOGRAPHIC CONCEPTUAL PROJECTION

Yilmaz Degirmenci
First Lieutenant Turkish Army
B.S. Military Academy, 1997

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
September 2002

Author: Yilmaz Degirmenci

Approved by: Neil Rowe
Thesis Advisor

John Hiles
Second Reader

Chris Eagle
Chairman, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This thesis discusses the designing of an architecture which mimics a human thought mechanism. The architecture is called a Holographic Conceptual Projection, which uses analogy and dynamic pattern matching combined with some natural-language understanding. Our main hypothesis is that we project our way of thinking into words and sentences which we manipulate when thinking verbally. This means we can exploit the structure of sentences to build an algorithm that models our thought mechanism. In our Holographic Conceptual Projection Architecture we give examples of every word within the context patterns. The patterns contain sentences that describe the "condition", "desired situation", "proposition" and "outcome" of the concept. The concept's patterns are then compared with new cases to see analogies. This comparison is done with dynamic generalization and specialization techniques. Finally after building an implementation, we tested it on an intelligent file-management system and an image-processing application.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I must express my deepest gratitude and appreciation to Professor Neil Rowe for his guidance, encouragement, and patience during this most challenging part of my education at the Naval Postgraduate School. I wish to thank Professor John Hiles for his valuable suggestions and motivational leadership. I am grateful to the Turkish Army for providing me with the privilege of studying at NPS.

My special thanks go to my friend Richard Woodmaster for being a beacon in all ways. I dedicate this research to my mother Esma Muruvvet Degirmenci for being a endless source of Love and Light.

THIS PAGE INTENTIONALLY LEFT BLANK

I: INTRODUCTION

Throughout history, many researchers from different disciplines have studied mind mechanisms trying to describe and systemize them. Today many researchers are exploring areas, such as the philosophy of mind, artificial intelligence, cognitive science, neuroscience, and psychology to solve this mystery. Among these areas, researchers are working to describe "verbal thinking".

For our research, we observed the mechanisms within our own mind before investigating and comparing our ideas with theories from other disciplines of study. We knew that if we could describe the process of our mind mechanisms during thinking, we could define it as an algorithm. Then using computer technology, we could code the process. Such an algorithm could easily be applied in any area of technology.

We discovered that there are around 300 basic verbal thinking concepts that can be the foundation for defining other concepts. For example, defining the concepts "good" and "bad" helps explain the words "wonderful", "excellent", "dreadful", or "ruined". Additionally, defining the concept "more" helps explain "better", "best", "worse" and "worst". This indicates that if we can build knowledge of a few key concepts, then it is possible to add and integrate most of a dictionary.

A second aspect we learned was that the human mind perceives nothing individually. That is, humans learn everything within structures relating to subject, object, location, time and action. That led us to describe every single word and object within a sentence-like structure. Then we could see that many concepts are connected to the environment by conditions. Many concepts involve a change or motion, a desired action or purpose, and an output. As an example, WHEN I'm at home, IF I'm hungry, THEN I eat food, SO I become full. Here "when" corresponds to a condition, "if" corresponds to a change, "then" to a desired action or proposition, and "so" corresponds to result.

Concepts are connected in a tree-like structure: a type hierarchy. In defining this hierarchy, we can use it to make generalizations and specializations about concepts. For example, “Mary drinks coke” and “John drinks soda” are structurally the same. Here both “Mary” and “John” are humans and both “Coke” and “soda” are beverages. People learn from examples experienced in life, from which they make generalizations. Therefore dynamic generalization and specialization must be a key feature in our algorithm.

We used “and” and “or” to improve the ability of each part of a pattern to describe a concept enabling us to define concepts of varying length and complexity. Thinking also addresses a variety of specialization levels. For example, “I ate calamari yesterday”, “I eat calamari”, “Humans eat calamari”, “Humans eat food”, and “Living things need energy” represent different levels and aspects of the same reality. We believe that if we define 300 basic concepts by describing them in patterns and using those patterns as templates to compare with new cases while using generalizations and specializations on these cases, we can get a wide range of common-sense information or conclusions. This whole mechanism can be called “Holographic Conceptual Projection Architecture” (HCPA). The smallest atomic element is “the clause”, and every single clause is capable of interfacing with any clause or concept within the whole system.

We can compare our model with case-based reasoning models. All case-based reasoners share a common way to address a new problem: retrieve appropriate cases from its memory, modify a retrieved case applicable to the current situation, apply the transformed case to the new problem, and save the solution with a record of its success or failure for future use. (Luger & Stubblefield, 1999, Section 6.4.1) However our approach will use reasoning by analogy rather than the similarity calculation of case-based reasoning. It will use a kind of forward chaining (Wallis & Moss, 1995) to make inference.

We divided our research into two phases. The first phase created a prototype program to demonstrate the ideas explained above using Delphi (Object Pascal). The second phase applied that algorithm to a real problem demonstrating the validity and practicality of the algorithm. Next we built a framework for knowledge of Java functions for image processing, which draws and recognizes images converting them to a 3-

dimensional model. This application requires only 50 to 100 concepts and shows the features and power of the algorithm.

Chapter II introduces some related work in artificial intelligence; each work describes a different aspect of analogy. Chapter III describes the design and the data structures used in our program. Chapter IV explains how the design works by using examples from a test program. Chapter V shows how concepts in this architecture can be associated to Java functions. Chapter VI describes how this work can be improved as a future work. Chapter VII gives the conclusions of this research.

THIS PAGE INTENTIONALLY LEFT BLANK

II: PREVIOUS WORK

A wide range of work is relevant to our study. In his dialogue *The Republic*, Plato introduced "the allegory of the cave and divided line." For Plato, human beings live in a world of visible and intelligible things. The "visible world" is what surrounds humans: what is seen, what is heard, and what is experienced. This visible world is a world of change and uncertainty. The "intelligible world" is composed of the unchanging products of human reason; anything arising from reason alone, such as abstract definitions or mathematics, makes up this intelligible world, which is the world of reality. The intelligible world contains the eternal "forms" (idea in Greek) of things; the visible world is the imperfect and changing manifestation in this world of these unchanging forms. For example, the "form" or "idea" of a chair is intelligible, abstract, applies to all chairs, and it never changes. "A chair is a piece of furniture consisting of a seat, legs, back, and often arms, designed to accommodate one person" (American Heritage Dictionary, 1992). However chairs can vary wildly among themselves, such as a chair with three or four legs, a chair with wheels, an armchair, or a rocker. An individual chair is a physical, changing object that can easily cease to be a chair (if, for instance, it gets broken); the form of a chair or "chairness" never changes. As a physical object, a chair only makes sense in that it can be referred to the "idea" of chairness. In our work a Concept Database holds such abstract definitions of concepts; whereas, an Event Database defines the current state of the world.

In psychology, human learning is divided into six general categories: conditioning, motor learning, discrimination learning, verbal learning, problem solving, and concept learning (Fogiel, 1999). Our work domain involves only the last three categories: verbal learning, problem solving, and concept learning of psychology. Learning verbal associations provides an important link between elementary non-verbal learning process, language, and thought. Human problem-solving is regarded as "thinking" with several stages: the problem is stated, evidence for a solution is arranged, an idea emerges, alternatives are evaluated, and the solution is verified. Concept learning involves attaching verbal labels to the phenomena of the world.

For Pinker (2000) irregular and regular forms in language are the outcome of two mental subsystems, “words” and “rules”, expressing an event or state that took place in the past with both being equally important. In this work, we define words and their hierarchies in a Root Database and rules to build sentence structures and concept blocks in other data structures. (Bernstein, 1977) notes, “You won’t find it in most dictionaries, but *flied* is the past tense of *fly* in one specialized field: baseball. You could not say of the batter who hoisted a can of corn to the center fielder that he “flew out”; you must say he “flied out”.” Therefore, we argue that every verb must be defined in a whole sentence-like structure containing the subject, object, location, time, and instrument defining the condition of that verb.

Conceptual Dependency Theory (Schank and Rieger, 1974) offers a set of four primitive conceptualizations from which it is claimed that the entire world of meaning or “semantics” is built:

ACTs	actions
PPs	objects (picture producers)
AAs	modifiers of actions (action aiders)
PAs	modifiers of objects (picture aiders)

All actions are assumed to be comprised of one or more of these primitive ACTs:

ATRANS	transfer a relationship (give)
PTRANS	transfer physical location of an object (go)
PROPEL	apply physical force to an object (push)
MOVE	move body part by owner (kick)
GRASP	grab an object by an actor (grasp)
INGEST	ingest an object by an animal (eat)
EXPEL	expel from an animal’s body (cry)
MTRANS	transfer mental information (tell)
MBUILD	mentally make new information (decide)

CONC	conceptualize or think about an idea (think)
SPEAK	produce sound (say)
ATTEND	focus sense organ (listen)

These primitives are used to describe the meaning of structures with case relations and other kinds of associations involving objects. Conceptual-dependency relationships are conceptual syntax rules constituting a grammar of meaningful semantic relationships. This theory argues that "ACTs" are the key elements to describe a concept. In our work we describe concepts in a Concept Database using verbs as the key elements to describe those concepts. We built connections among verbs, subjects, objects, and other elements by defining them within a "clause structure."

Carbonell (1983) proposes a theory of logical problem-solving using analogy. It outlines a "logical transformation process" that is developed to extract knowledge from past successful problem-solving situations bearing a strong similarity to the current problem. The theory expands standard "means-ends analysis" with a reminder and transformation mechanism. The reminder mechanism exploits the knowledge of solutions to previous problems by comparing the differences in the initial and final state, the path constraints, and the operator preconditions of the present and previous problem spaces. The other mechanism transforms the old solution sequence into one that satisfies the criteria of the new problem. As an example, the paper explains the monkey-and-bananas and experimenter-and-bananas problem from the viewpoint of the analogical problem-solving model:

> A monkey watches a behavioral psychologist pick up a wooden box and place it under a hook in the ceiling. Next, the experimenter climbs on the box, places some bananas on the hook, climbs off the box, and returns the box to its original location. Then, the experimenter releases the (hungry) monkey and leaves the room. Can the monkey benefit from having observed the experimenter?

From the point of view of analogical problem-solving, the monkey's problem is "initial state" = monkey on the floor, bananas on the ceiling, box in the room; "final state" = monkey in possession of the bananas; "path

constraints" = physical abilities of the monkey. However, the solution to the experimenter's decision will not directly help the monkey.

At first the monkey was able to use standard means-ends analysis to solve the problem (compare the current state to the goal state, choose an operator that reduces the difference, apply the operator if possible, if not solve a sub-problem first and then resume work on the original problem). Therefore, the monkey who could select the operator GET-OBJECT applied to bananas. This operator suffers an unsatisfied precondition: The monkey cannot reach the bananas. As a result, the active subgoal becomes to reach the ceiling where the bananas are located. If the monkey recalls the observation of the experimenter, it may realize that the problem of reaching the ceiling has already been solved. The monkey may apply the "parameter-substitution T-operator" (substituting "monkey" for "experimenter") and, optionally, the "solution-sequence truncation T-operator" (eliminating the need to return the box to its original location after having used it). This problem-solving process in the "T-space" results in a plan that the monkey can apply directly to reach the bananas. Our work uses a similar approach of "logical transformation" to transform from previously defined cases in a Concept Database to newly encountered cases and to update the current state of the world according to that new case. Nevertheless our work addresses inference and not planning.

As an example of a quite different "connectionist" approach to analogy, Mitchell (1993) is a model based on the premise that analogy-making is fundamentally a high-level perceptual process in which the interaction of perception and concepts give rise to "conceptual slippages" that allow analogies to be made. With the strategy of "isolate and idealize", that approach is applied to a computer model called Copycat (Hofstadter, 1984). In Copycat, both concepts and high-level perception are emergent phenomena, arising from large numbers of low-level, parallel, non-deterministic activities. A sample problem in a Copycat computer model might be "abc => abd → ijk => ?". Here, "abc" is the "initial state", "ijk" is the "target state", and "abd" is the "modified string". By using the program, the operator supposingly discovers the "same way" to come up with a reasonable result.

There are four mechanisms within a Copycat computer model:

1) The "Slipnet" is a network of nodes where concepts such as A-Z, 1-5, *left, right, sameness*, and etc. reside. A node is activated when instances of it are perceived by "codelets" such as a "modified-string replacement for the "b" in "abc", a "bond" from the "j" to the "k", and a "correspondence" between the "c" and the "k"." During a run of the program on a given problem, the probability that a node will be brought in or be considered further by codelets is a function of the node's current activation level. Thus, there is no black-and-white question of whether a given concept is consciously used at a given time; continuous activation levels and probabilities allow different concepts to be present to different degrees.

2) In addition to the Slipnet, where long-term concepts reside, the "Workspace" is another data structure, in which perceptual structures are built hierarchically on top of the "raw" input (the three strings of letters). For example, "leftmost" as a description of "a" in "abc", a "successorship" bond between "a" and the "b" in "abc" are some structures defining the relationships among concepts.

3) Codelets are stored in "Coderack." Any run starts with a standard initial population of bottom-up codelets (with preset urgencies changing by probabilistic choices) on the Coderack.

4) A final mechanism "temperature" measures the degree of perceptual organization in the system and controls the degree of randomness used in making decisions.

Since the program is permeated with non-determinism, different answers arise on different runs. However, although every run is different at the microscopic level, statistics lead to far more deterministic behavior at the macroscopic level. This notion of microscopic non-determinism resulting in macroscopic determinism suggests many

useful features for future study. However, our study focuses on simpler deterministic mechanisms for analogy in analyzing their capabilities and limits.

III: DATA STRUCTURES FOR OUR INFERENCE PROGRAM

Our objective is to provide a word-centered mechanism reflecting the reality within a world of words. Humans can use and learn any system primarily by analogy techniques. However, they still do not have a general computer algorithm for analogies. We think the only possible way is to describe examples of verbal thinking in structures made of words. This is a natural-language approach.

Humans have a memory where information and definitions of concepts are stored; they have sense organs, such as ears, eyes and skin to get information from the outside and organs, such as mouth and hands to act or communicate. All these mechanisms are about perceiving, saving, analyzing and manipulating information. For the sake of simplicity and ease of the model, we divided a data structure representing memory into three parts: a Root Database, a Concept Database, and an Event or Real World Database. Additionally we have Input (Listen) and Output (Talk) data structures to receive and post information.

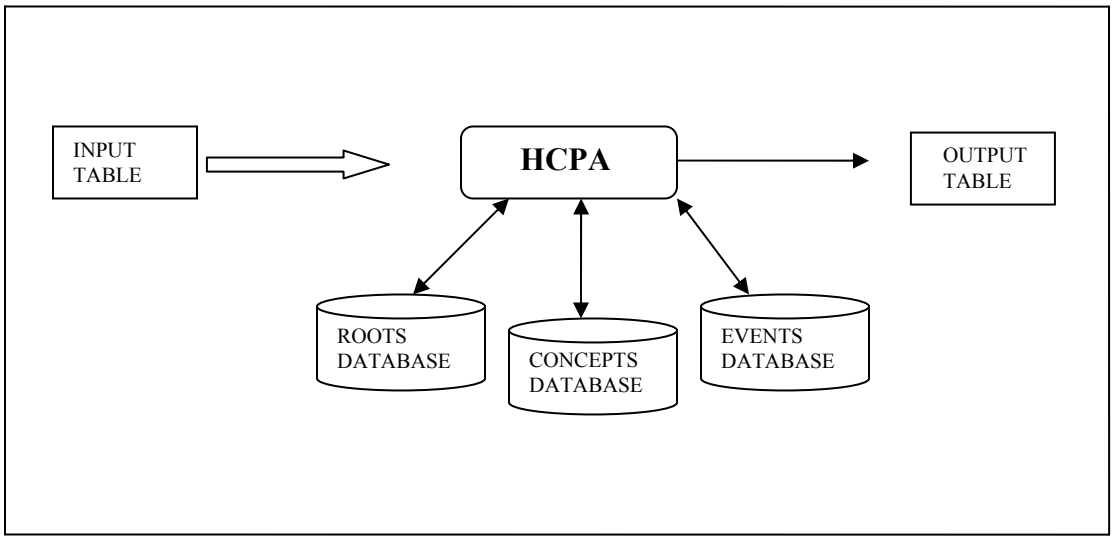


Figure 3.1 Data Structures in HCPA

Root Database: This is a data structure for words and their roots. If a word is logically implied by the meaning of another word or is a "supertype" of the word, it is the root of that word. The ultimate root is "concept", so all the other words in the program are "subtypes" of concept. For example, a classroom is a room, a room is a location, and location is a concept; and to learn is a verb, which is a concept. A word can have only one root in our system, so no multiple inheritance is allowed. Word hierarchies permit generalizations, specializations, and comparisons of concepts. Below are some root-branch pairs:

ROOT DATABASE:

ROOT	BRANCH		ROOT	BRANCH		ROOT	BRANCH		ROOT	BRANCH
concept	matter		matter	living-thing		location	room		time	day
concept	verb		living-thing	plant		location	kitchen		time	now
concept	time		living-thing	animal		room	classroom		location	home
concept	location		animal	human		room	bathroom		verb	enjoy
concept	property		human	Yilmaz		verb	eat		verb	possess
concept	means		human	Jason		verb	drink		verb	have
concept	event		human	sister		matter	water		verb	has
concept	adjective		human	mother		water	beverage		human	friend
concept	particle		human	father		beverage	Coke		verb	be
concept	energy		human	I		beverage	Fanta		verb	get
concept	adverb		matter	food		verb	live		matter	object
concept	value		food	fruit		verb	change		verb	use
concept	quantity		verb	move		matter	stone		conjunction	If
concept	relation		move	go		verb	take		conjunction	then
concept	conjunction		go	walk		verb	learn		conjunction	when
color	orange		move	come		property	number		location	library
fruit	orange		property	color		adjective	thirsty		location	school

Table 3.1 Root Database

Like WordNet (Miller, 1993), our data structure allows a word to have multiple meanings. For example orange can represent both a color and a fruit. We used our own data structure instead of WordNet to be able to manipulate the data structure freely. Nonetheless we can still use WordNet to build our own lexical data structure.

Concept Database: This data structure holds concept structures. We call each row a "clause." As explained in the Introduction, each concept can have a condition clause (when), a change-of-situation or goal-state clause (if), and a conclusion or desired-

action clause (then). The "if" clause is like a goal state for that concept. The "when" clauses define the necessary conditions for that goal state to happen. The "then" clause defines the necessary action to obtain that goal state. The structure of concepts is a special case of the *if... then...* rules in a rule-based expert system (Luger & Stubblefield, 1999). For each clause, we optionally identify subject, verb, object, location, time, conjunction, and instrument "cases" (Allen, 1983). For example in the sentence "Today I wrote a poem in the bookstore with my pen", "I" is the subject, "wrote" is the verb, "poem" is the object, "bookstore" is the location, "today" is time, and "pen" is the instrument. Like in a "means-ends analysis", we use cases to explain the differences between two sentences. Below is a sample table that defines 5 concepts:

CONCEPT DATABASE:

	SUBJECT	VERB	OBJECT	PLACE	TIME	CONJUNC	INSTRUMENT
1	I	be		classroom		when	
1	Jason	be		classroom		when	
1	Jason	be		library		if	
1	Jason	go		library		then	
2	I	be		classroom		when	
2	Jason	be		library		when	
2	Jason	be		classroom		if	
2	Jason	come		classroom		then	
3	I	be		classroom		when	
3	I	be		library		if	
3	I	go		library		then	
4	I	get thirsty				if	
4	I	drink	water			then	
5	I	enjoy	taste			if	
5	I	drink	coke			then	

Table 3.2 Concept Database

The first concept above (lines 1 thorough 4) is an example of the concept "go," translated as: "When I'm in the classroom and Jason is in the classroom, if Jason changes his location so he is in the library, that means Jason goes to the library." The second concept exemplifies the concept "come", the opposite of "go." This is translated as: "When I'm in the classroom and Jason is in the library, if he changes his location and he is in the classroom now, that means Jason comes to the classroom." The third concept

also exemplifies "go" when I change my location or someone else changes his or her location without referencing me. A simple translation is: "When I'm in the classroom, if I change my location and I'm in the library now, that means I go to the library." The fourth and fifth examples can be paraphrased as: "If I get thirsty, I drink water", and "If I drink coke, the reason for that is I enjoy its taste."

We can relate the columns of our Concept Database to the case relations in natural-language understanding (Allen, 1983). With case relations, sentences with different syntactic structures, but with the same meaning, should get mapped to similar structures. For example, consider the sentences "John broke the window with a hammer," "the hammer broke the window," and "the window broke." John, the hammer, and the window play the same semantic roles in each of these sentences. John is the actor, the window is the object, and the hammer is an instrument used in the act "breaking of the window."

Event – Real World Database: This data structure holds the information of the current state of the world. The clauses here will be matched with the clauses of the Concept Database. An example is the following:

SUBJECT	VERB	OBJECT	PLACE	TIME	CONJUNC	INSTRUMENT
I	be		home			with my family
my father	be		bathroom			
my mother	be		kitchen			
my sister	be		school			
she	drink	Fanta				
we	live		Monterey			

Table 3.3 Event Database

Input – Listen Table: This is the data structure where the new goal is described. This is imagined as the "if" sentences or goal statements of the new case.

Output – Talk Table: The program generates a proposition or desired action for the goal state entered in the Input Table and displayed in the Output Table. This table works as a query for a special form of backward chaining like in Prolog (Atkin, 1999).

The major drawback of our data structure design was that it did not initially permit giving more than one word for each column within a clause. (In Chapter V we

improved our design so that it permits entering more than one word for each column within a clause.) This drawback made it impossible, for example, to refer details of the taste to Coke or to use more complex structures such as negation or quantified variables. On the other hand, it allowed a representation of approximately 98 % of natural-language phenomena.

THIS PAGE INTENTIONALLY LEFT BLANK

IV: DESCRIPTION OF THE PROGRAM

Our implementation of HCPA is in the form of a program that does pattern-matching between the information in the memory (Concept Database), the goal information (Input Table), and the current world state (Event Database). If certain matches occur, the program comes up with new inferences and conclusions. We can relate our design to the design of Prolog: the query in Prolog is input, facts are the Event Database, and rules are the Concept Database in our design. However our inference method is different since it uses analogy. The whole system, referred to as "projection", updates the information in the memory for a new case.

The following represents the present level of development for HCPA:

1) Can accept new inputs and project those changes to the Event Database and to the Output Table.

2) Is able to monitor an up-do-date Event Database that reflects the present state of objects in the system by making inferences according to the information in the Event Database and updating that information according to the Output Table after making inference.

3) Can be incorporated in other software to provide a form of natural-language understanding that includes projection.

The best way to understand how the program works is explaining with examples: Let us assume that five concepts are defined in the Concept Database:

CONCEPTS DATABASE

	SUBJECT	VERB	OBJECT	PLACE	TIME	CONJUNC	INSTRUMENT
1	I	be		classroom		when	
1	Jason	be		classroom		when	
1	Jason	be		library		if	
1	Jason	go		library		then	
2	I	be		classroom		when	
2	Jason	be		library		when	
2	Jason	be		classroom		if	
2	Jason	come		classroom		then	
3	I	be		classroom		when	
3	I	be		library		if	
3	I	go		library		then	

4	I	get thirsty				if	
4	I	drink	water			then	
5	I	enjoy	taste			if	
5	I	drink	Coke			then	

Table 4.1 Concept Database

Next, let us suppose the following information in the Event Database:

EVENT DATABASE

SUBJECT	VERB	OBJECT	PLACE	TIME	CONJUNCTION	INSTRUMENT
I	be		home			with my family
father	be		bathroom			
mother	be		kitchen			
sister	be		school			
sister	drink	Fanta				

Table 4.2 Event Database

What follows are sample cases of our implementation. Each case shows a different feature of our design. Our approach can be described as follows:

For an input case (Input Table), find proper proposition(s) for that case (to be displayed in the Output Table) by comparing it with pre-defined concepts in the memory (Concept Database), determining a match concept from them, obtaining the necessary information related to that match concept as its present situation in the current state of the world (Event Database), and projecting the proposition of that concept as the new proposition. Then display that new proposition in the Output Table and update the Event Database with the information coming from the Input Table.

The program matches clauses by comparing their corresponding sections. For example the sentences “I play soccer” and “Chris plays basketball” match because I and Chris are both humans, and both soccer and basketball are games. The program uses the Root Database to determine if the corresponding words have the same root or not (if they share a common root before the level of concept), but the objects cannot move down for further comparison. For example, “cooking pizza” means “cooking food” at the same time, but “cooking food” does not always mean “cooking pizza.” The verb is the most important attribute of a clause because it defines what that clause as a concept means;

therefore verbs must be the same to match. We can match the clause “I drink water” with the clause “my mother drinks coffee”, but we cannot match the clause “I drink water” to the sentence “I write poem.”

A. CASE 1: PROJECTION

1) Assume input clause from the Input Table is

INPUT TABLE:

	SUBJECT	VERB	OBJECT	PLACE	TIME	CONJUNC	INSTRUMENT
	sister	be		home			

Table 4.3 Input Table

(See Figure 4.1 for a diagram of the complete set of data structures for this case)

2) Find matching “if” clauses in the Concept Database for that input clause.

- “Jason be library” (Concept Block 1)
- “Jason be classroom” (Concept Block 2)
- “I be library” (Concept Block 3)

Matching here means that, with the exception of the verbs (which are critical), the words in each section of the input clause are either the same or share the same root with the words in the same section of the corresponding “if” clause. For example the “if” sentence of Concept Block 1 is “Jason be library.” Both “sister” and “Jason” are humans; both “home” and “classroom” are locations, and both their verbs are “be.” Therefore the input clause and that “if” clause match. The program uses the Root Database to determine whether two words have the same root (before the level of “concept”) or not.

3) Determine which of the concept blocks have conditions that match their “when” clauses to the Event Database. In our example, we now have three potentially matching concept blocks in the Concept Database. At first the program tries the Concept Block 1:

a) Create new condition clauses by replacing the subjects of “when” clauses with the subject of the input clause:

The second clause of the Concept Block 1 (Jason be classroom) becomes "sister be classroom."

b) Match "when" clauses to clauses to in the Event Database. Subjects here are critical in terms of matching because the program uses them to connect to the current state of the world. Therefore, for two clauses to match, both their "subjects" and "verbs" must be the same

In our example, "I be classroom" matches to "I be home" (clause 1) and "sister be classroom" matches to "sister be school" (clause 4).

c) Build a new concept block for current situation.

- "When I be home" (from the Event Database)
- "When sister be school" (from the Event Database)
- "If sister be home" (from the Input Table)

d) Number the words of both the new concept block and the old concept block from the Concept Database:

NEW CONCEPT BLOCK:

	SUBJECT	VERB	OBJECT	PLACE	TIME	CONJUNC	INSTRUMENT
	I	be		home		when	
	sister	be		school		when	
	sister	be		home		if	
	?	?		?		then	
	1			**2**			
	3			**4**			
	3			**2**			

Table 4.4 Concept Block

CONCEPT BLOCK 1 (FROM CONCEPT DATABASE)

	SUBJECT	VERB	OBJECT	PLACE	TIME	CONJUNC	INSTRUMENT
1	I	be		class		when	
1	Jason	be		class		when	
1	Jason	be		library		if	
1	Jason	go		library		then	

	1			**2**			
	3			**2**			
	3			**4**			

Table 4.5 Concept Block

Here the pattern of the Concept Block 1 does not match the pattern of the new concept block. But the program will return to step "a" to try the next matching concept block (Concept Block 2) from the Concept Database. Because its pattern is the same as the pattern of the new concept block (see Table 4.6), the program continues to the next step.

CONCEPT BLOCK 2 (FROM CONCEPT DATABASE)

	SUBJECT	**VERB**	**OBJECT**	**PLACE**	**TIME**	**CONJUNC**	**INSTRUMENT**
2	I	be		classroom		when	
2	Jason	be		library		when	
2	Jason	be		classroom		if	
2	Jason	go		classroom		then	
	1			**2**			
	3			**4**			
	3			**2**			

Table 4.6 Concept Block

e) If the pattern match is successful, generate a new proposition by replacing the words in the "then" clause of the matching concept block from the Concept Database with the corresponding words from the new concept block:

The "then" clause of the Concept Block 2 is "Jason be classroom". The number of "Jason" in the pattern is 3, and the number of "classroom" is 2. Number 3 in the new concept block represents "sister", and number 2 represents "home." As a result, the "then" clause of the new concept block becomes "sister come home."

4) Display the generated proposition in the Output Table and update the Event Database for the present situation:

OUTPUT TABLE:

	SUBJECT	VERB	OBJECT	PLACE	TIME	CONJUNC	INSTRUMENT
Case	sister	is		home			
Proposition	sister	come		home			

Table 4.7 Output Table

EVENT DATABASE

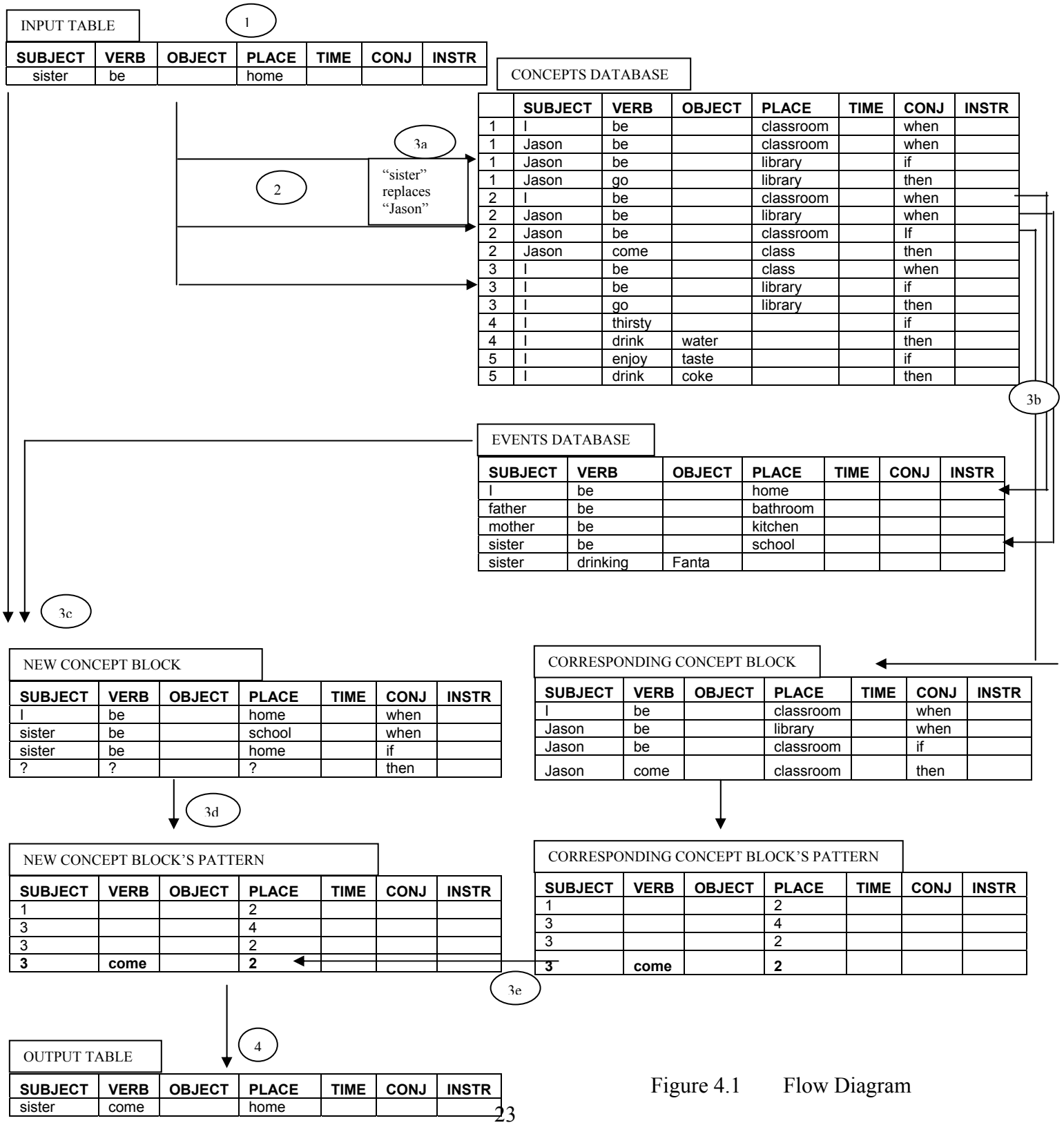
SUBJECT	VERB	OBJECT	PLACE	TIME	CONJUNC	INSTRUMENT
I	be		home			with my family
father	be		bathroom			
mother	be		kitchen			
sister	be		home			
sister	drink	Fanta				

Table 4.8 Event Database

If the input were "sister be school" in the Input Table and "sister is home" in the Event Database, the corresponding concept block would be Concept Block 1 and the Output would be "sister go school" in the Output Table and "sister is school" in the Event Database.

If the input were "sister be library" in the Input Table and "sister is school" in the Event Database, the corresponding concept block would be Concept Block 3 (because there is no reference to me), and the output would be "sister go library" in the Output Table and "sister is library" in the Event Database.

FLOW DIAGRAM



SUBJECT	VERB	OBJECT	PLACE	TIME	CONJ	INSTR
sister	be		home			

	SUBJECT	VERB	OBJECT	PLACE	TIME	CONJ	INSTR
1	I	be		classroom		when	
1	Jason	be		classroom		when	
1	Jason	be		library		if	
1	Jason	go		library		then	
2	I	be		classroom		when	
2	Jason	be		library		when	
2	Jason	be		classroom		If	
2	Jason	come		class		then	
3	I	be		class		when	
3	I	be		library		if	
3	I	go		library		then	
4	I	thirsty				if	
4	I	drink	water			then	
5	I	enjoy	taste			if	
5	I	drink	coke			then	

SUBJECT	VERB	OBJECT	PLACE	TIME	CONJ	INSTR
I	be		home			
father	be		bathroom			
mother	be		kitchen			
sister	be		school			
sister	drinking	Fanta				

SUBJECT	VERB	OBJECT	PLACE	TIME	CONJ	INSTR
I	be		home		when	
sister	be		school		when	
sister	be		home		if	
?	?		?		then	

SUBJECT	VERB	OBJECT	PLACE	TIME	CONJ	INSTR
I	be		classroom		when	
Jason	be		library		when	
Jason	be		classroom		if	
Jason	come		classroom		then	

SUBJECT	VERB	OBJECT	PLACE	TIME	CONJ	INSTR
1			2			
3			4			
3			2			
3	**come**		**2**			

SUBJECT	VERB	OBJECT	PLACE	TIME	CONJ	INSTR
1			2			
3			4			
3			2			
3	**come**		**2**			

SUBJECT	VERB	OBJECT	PLACE	TIME	CONJ	INSTR
sister	come		home			

Figure 4.1 Flow Diagram

B. CASE 2: HIERARCHY IN PROJECTION

This case illustrates another feature of HCPA: Some concepts may share the same results at different levels of detail. In addition, this case also runs the projection algorithm for "then" clauses of concept blocks to accomplish a kind of backward chaining.

1) Assume the input clause from the Input Table is

INPUT TABLE:

	SUBJECT	VERB	OBJECT	PLACE	TIME	CONJUNC	INSTRUMENT
	sister	drink	Fanta				

Table 4.9 Input Table

2) None of the "if" clauses in the Concept Database matches the input clause; however two "then" clauses match.

- "I drink water" (Concept Block 4)
- "I drink Coke" (Concept Block 5)

Because both "Fanta" and "Coke" are subtypes of "beverage", they match. Because "Fanta" is a subtype of "water", they also match (however if Fanta were a supertype of water, they would not match). "I" and "sister" are both humans, so they match.

3) Determine which of these concept blocks have "when" conditions that match clauses in the Event Database. Because both Concept Block 4 and Concept Block 5 have no "when" clauses defined in the example, the program skips this step.

4) Build a new concept block for the new case by updating the old concept block from the Concept Database (the program updates the old concept block by numbering each word in that concept block and projecting that number pattern to the new case as we explained in Case 1):

- "if sister (gets) thirsty, then sister drink(s) water"
- "if sister enjoy(s) (its) taste, then sister drink(s) Fanta"

5) Display the results in the Output Table (if the conclusion clause is an "if" clause, it is interpreted as a "proposition"; if it is a "then" clause, it is interpreted as a "reason" for the "case").

OUTPUT TABLE

	SUBJECT	VERB	OBJECT	PLACE	TIME	CONJ	INSTRUMENT
Case	sister	drink	Fanta				
Reason	sister	thirsty					
Reason	sister	enjoy	taste				

Table 4.10 Output Table

If the input case were "sister drink water", the only corresponding concept block would be the Concept Block 4 and the only output would be "sister (gets) thirsty" (water is a supertype of Coke, therefore they do not match).

The more attributes a goal-state has, the more sophisticated the desired actions are. For example, in the situation "I get hungry" a conclusion could be simply "I eat food". On the other hand, for the situation "I am hungry at school during morning" a conclusion could be "I'm eating food in lunch-room"; and for the situation "I am hungry at school during evening" it could be "I'm eating pizza in cafeteria". Note that we use the present progressive tense to define specific situations of specific people, whereas we use the simple present tense to define general situations. (We used in our test program only examples of simple present tense.)

THIS PAGE INTENTIONALLY LEFT BLANK

V: ASSOCIATING CONCEPTS WITH JAVA FUNCTIONS

Another application explored in our program was an intelligent window-based image-processing program. We added the following new features to the architecture to accomplish this task:

- Associating concepts with Java functions.
- Distinguishing instances from types within the Root Database so as to increase vocabulary.
- Handling multiple words within each column entry.
- Providing a triggering mechanism among concepts.
- Updating the current state of the world by updating the Event Database.

The tables below explain each of these new features with the necessary data structures for this program:

ROOT DATABASE:

ROOT	TYPE	INSTANCE	ROOT	TYPE	INSTANCE
concept	memory		verb		delete
concept	verb		verb		is
concept	particle		verb		are
concept	conjunction		verb		draw
concept	user		verb		paint
concept	shape		user		User1
concept	color		user		Yilmaz
concept	function		conjunction		and
memory	drive		conjunction		not
drive	directory		conjunction		if
drive		C	conjunction		then
drive		D	conjunction		when
directory		Dir1	particle		in
directory		Dir2	particle		from
shape	point		particle		to
shape	line		particle		for
shape	circle		particle		above
shape	square		particle		below
point		Point1	particle		all
line		Line1	particle		none
circle		Circle1	function		OpenFile
square		Square1	function		DrawLine

color		yellow	memory	File	
color		red	file	textFile	
color		blue	file	imageFile	
verb		open	file		File1
verb		close	file		File2
verb		copy	point		20,20,70,70
verb		cut	point		10,10,50,50
verb		create			

Table 5.1 Root Database

We divided "branch" in the Root Database into "type" and "instance." For example, "color" is a type, whereas "red" or "blue" are instances of that type. Distinguishing instances from types more precisely defines the words and their relationships with each other. Only instances can be actively used within the system; types are helpers to recognize an instance. For example, a person cannot input "User1 open File" as a goal state because "File" is not an instance; however if a person enters "User1 open File Lake.jpg", the program can infer that "Lake.jpg" is an instance of type "File", allowing its information to be added into the Root Database even if it has not been defined before. Hence, only "ground instances" can be used to define "if" and "then" clauses in the Concept Database.

CONCEPT DATABASE

	SUBJECT	VERB	OBJECT	PLACE	TIME	CONJUNC	FUNCTION
1	User1	open	File1	in Dir1 in C		if	
1	User1		File1	in Dir1 in C		then	OpenFile
1	drive	is	C			so	
1	directory	is	Dir1			so	
2	directory	is	Dir1			when	
2	drive	is	C			when	
2	User1	open	File1			if	
2	User1	open	File1	in Dir1 in C		then	
3	User1	draw	Line1	for 10,10,50,50 and for red		if	
3	User1		File1	for 10,10,50,50 and for red		then	DrawLine
3	color	is	red			so	
3	point	is	10,10,50,50			so	
4	color	is	red			when	
4	point	is	10,10,50,50			when	
4	User1	draw	Line1			if	
4	User1	draw	Line1	for 10,10,50,50 and for red		then	

Table 5.2 Concept Database

In this program we used “Function” as the last column of the Concept Database instead of “Instrument”. Functions are used to associate a “then” clause to a Java function; therefore, they can be found only in a “then” clause of a concept block. Additionally, we added “so” as a new kind of clause into our concept block structure (in Concept Block 1 and in Concept Block 3) to update the Event Database.

Concept Block 1 defines (lines 1 to 4) the concept “to open a specific file.” This is translated as: “If a user (“User1”) opens a file (“File1”) in a directory (“Dir1”) and on a drive (“C”), then the program uses the OpenFile function with those parameters. Then the drive information is updated as “C” along with updating the directory information as “Dir1” in the Event Database (by using “so” clauses). Concept Block 2 utilizes the information Concept Block 1 generates. The “then sentence” of Concept Block 2 is the same as the “if sentence” of Concept Block 1. This is translated as: “When the directory information in the Event Database is “Dir1” and the drive information is “C”, and the command is “open the file File1”, it is understood as "open the file “File1” in directory “Dir1” and in directory “Dir2”.” Because this is a function-based application, the program must activate a Java function in order to stop. Because the “then” clause of Concept Block 2 does not contain a “Function”, the program uses the “then” clause as the input case of a new iteration (which is the same as Concept Block 1). Thus Concept Block 2 “triggers” Concept Block 1 while Concept Block 1 “triggers” the necessary Java function (See Case 4 for further explanation of the “triggering mechanism”). Concept Block 3 and Concept Block 4 use similar ideas to draw a line for a specific point and color value.

The Event Database shows the “active” drive, directory, color, and point values used in the system. Whenever a new file is opened or a new shape is drawn, its information is reflected in the Event Database.

EVENT DATABASE

SUBJECT	VERB	OBJECT	PLACE	TIME
drive	is	C		
directory	is	Dir1		
color	is	red		
point	is	10,10,50,50		

Table 5.3 Event Database

A. CASE 3: ABILITY TO USE FUNCTIONS AND TO INCREASE VOCABULARY

This case illustrates how the program generates a proper Java function for an input case by making the necessary inference. The program is able to accept new words not initially defined in the Root Database, (thus it can increase its vocabulary), and to update the Event Database.

1) Assume the input case from the Input Table is

INPUT TABLE

SUBJECT	VERB	OBJECT	PLACE
Yilmaz	open	file spiral.jpg	in directory thesis and drive C

Table 5.4 Input Table

The program parses each section of the input clause into individual words to manage them. However the program does not include during parsing a word, such as "file", "directory", or "drive", which is a "type" rather than an "instance" of a root. The program uses the "type" information to update the Root Database; for our current case, the words "spiral.jpg" and "thesis" do not exist in the Root Database, so the program adds the "spiral.jpg" as an instance of "file" and the "thesis" as an instance of "directory" into the Root Database.

2) Find matching "if" clauses in the Concept Database. (Because this is a function-based application, the program runs the projection algorithm only on "if" clauses and not on "then" clauses). For our case, Concept Block 1 is the matching concept block to the input clause.

3) Determine which of those concept blocks have conditions that match the current state of the world. In our case we have only one matching concept block, which does not have a "when" clause. Therefore the program skips this step.

4) If the "Function" attribute of that concept block is empty, use its "then" clause as the new input case and return to step 1; otherwise, trigger that function by utilizing the information within that "then" clause as the parameter information for that function.

Because the "Function" attribute of Concept Block 1 is not empty (OpenFile), the program triggers that function with its parameter values. Here the function gets its

parameter values from the “then” clause to create the path information for that file (C:\\thesis\\spiral.jpg).

5) Update the Event Database with the “so” clauses in the matching concept block. The program generates new condition clauses by substituting “drive is C” and “directory is Dir1” (the “so” clauses of Concept Block 1) with “drive is C” and “directory is thesis” (“C” and “thesis” are the active drive and directory names for this case) by utilizing their patterns.

B. CASE 4: TRIGGERING MECHANISM

This case illustrates how the output of an input case can be an input case for another situation. If the “Function” attribute in the Output Table is empty, that output clause is accepted as the new input clause of the Input Table.

1) Suppose the Input clause is:

INPUT TABLE

SUBJECT	VERB	OBJECT	PLACE
Yilmaz	open	spiral.jpg	

Table 5.5 Input Table

2) Find matching “if” clauses in the Concept Database.

The only matching concept block for that input clause is Concept Block 2.

3) Determine which concept blocks whose conditions exist in the Event Database.

The “when” clauses of Concept Block 2 are “directory is Dir1” and “drive is C”. Each has a corresponding clause in the Event Database. This means that Concept Block 2 matches the current situation.

4) If the “then” clause of that concept block contains “function”, trigger this function, or else use the updated “then” clause as the new input clause and return to the step 1.

Here the “then” clause of Concept Block 2 does not contain “function”; therefore, the program uses the “then” clause to build the new input clause. The new input clause

becomes: “Yilmaz open spiral.jpg in thesis in C”, which is, as we already know, the corresponding input clause of Concept Block 1. So, Concept Block 2 triggers Concept Block 1.

Similarly for the input:

INPUT TABLE

SUBJECT	VERB	OBJECT	PLACE
Yilmaz	draw	Line1	for point 30,40,150,100 and for color blue

Table 5.6 Input Table

The program will use Concept Block 3 to draw a blue line for the point values updating the Event Database as “color is blue” and “point is 30,40,150,100”. After that, entering the command “Yilmaz draw Line1” will match Concept Block 4 which triggers Concept Block 3 to draw a line.

If we had implemented this program by just adding the associative element “of” (such as “color of Line1” or “directory of Dir1”), we would have defined and kept track of multiple elements within the system. For example, the Event Database might have said: “Color of Line1 is Blue, Color of Circle1 is Red.” Similarly we were able to use the conjunction “and” to define multiple operations at once, such as: “User1 draw Line1 and Circle1.”

C. SEEING INTELLIGENTLY

The goal of our image-processing application is a program capable of drawing and recognizing shapes on a picture and converting them into 3-D models by associating shapes to concepts in a highly sophisticated way; however, we only had time to prepare a design for such a project. Initially we defined all the necessary words:

Verbs: Draw, paint, find, add, divide, have, be, join, intersect, read.

Objects: Region, area, surface, point, pixel, line, arc, edge, square, rectangle, box, cylinder, endpoint, jointpoint, part, piece, shape.

Properties and Particles: Straight, round, length, width, height, color, number, angle, all, none, same, different, true, false, more, and, or, not, of, at, light, dark, equal, thick, thin, wide, narrow, closed, perpendicular.

Locations: Bottom, top, right, left, over, beneath, between, inside, outside, side, next to.

Questions: What, where, how, which, how many, whose.

Values: Number, angle, distance and scale.

In order to explain the concepts to a computer program, we describe every concept in plain English first:

- To draw a point: To assign color to a specific pixel.
- To draw a line: To paint adjacent pixels between two points. A line has a thickness of a particular number of pixels.
- To draw a straight line: The angle between adjacent pixels is always the same.
- To draw an oblique line: The angle between adjacent pixels is varying.
- To draw a circle: To paint all pixels at the same distance of a particular point.
- To draw an arc: To draw a piece of circle.
- To find a line: To find pixels whose adjacent pixels in one direction are in the same color.
- To find surface: To find all adjacent pixels with the same color.
- To find angle: To find about the pieces of circle when two lines intersect.
- To draw a shape: To draw an object with its specific name, length, width, height, color and size values. To retrieve the information of the edge, the bottom or the top of a shape and the distance between shapes.
- To intersect: To have two shapes share the same pixels.
- To find the edge of a shape: To find the line on which adjacent pixels have different colors.

After describing these concepts, we can start building concept blocks to describe this world of drawings. Below are some examples:

If I draw *square1*
Then I draw line1 and
Then I draw perpendicularly line2 on line1's endpoint and
Then I draw perpendicularly line3 on line2's endpoint with line1's opposite direction and
Then I draw perpendicularly line4 on line3's endpoint with line2's opposite direction and
Then I have a closed shape
So square1 has four equal lines and
line1 is *part of* square1

If line1's color *is not* red
Then line1's color is blue (has a different color value in the pattern)

If line1's color is blue and
line2's color is blue
Then their color is *same*

If I draw line1 and
I draw line2
Then I draw *two* lines

The challenging part of this project is to integrate all the pieces of concepts into a whole picture. What is needed is a more complicated inference structure and triggering mechanism. For example, at the end, we want the program to be able to answer this question "If I draw a square then I have four lines; how many lines do I have, if I draw a triangle?" The program must be able to connect the square, triangle and math concepts to get the correct answer. Thinking is, in a sense, only asking the proper questions at the proper level.

Ultimately we hope the program will do the following jobs:

- Draw a 5cm long red line from pixel1 to pixel2.
- Find all lines whose color is not blue and whose length is greater than 3 cm.
- Draw square1 at the center.

- Draw cylinder1 below square1.
- Find the squares on the picture.
- Find the boxes on the picture and retrieve 3-D values.
- Find the cylinder1 on the picture, retrieve its 3-D values and scale it to double size.
- Find the human face on the picture; convert it into 3-D model by comparing and mapping it onto the default 3-D face model in the memory. (Humans can see a two- dimensional picture of a human being on a newspaper and imagine it three-dimensionally in their minds).

THIS PAGE INTENTIONALLY LEFT BLANK

VI: FUTURE WORK

This work is about an architecture mimicking human's way of thinking by analogy as closely as possible. In order to achieve this goal, we could add the following features to our system.

1) Similar to Carbonell's Logical Transformation introduced in Chapter II, dividing a main goal into subgoals and associating them with the main goal with a more complicated association and triggering mechanism among concepts.

2) A probabilistic decision making mechanism by defining the Java functions in a probabilistic way similar to the Copycat computer model explained in Chapter II. If one drew five circle shapes onto a paper, none of them would be exactly the same; we are not thinking in a black-and-white way.

3) As another aspect, making a decision means choosing the most optimum and valuable option among others. Our algorithm could have a specific value system to define the importance of each new case. We are already doing some of this in our daily statements: "I *like* watching TV, but I *have to* study for my exam tomorrow (Because if I am not successful on the exam, I will be *unhappy*). Here "like" shows pleasure, "have to" shows importance, and "unhappy" shows the result. Before offering a conclusion, we ask the question "why" at each step until we reach a result. To add this functionality to our algorithm, we must add another section with the name "because" into our block structure.

4) A holographic database mechanism to store information. We could build a structured knowledge around a seed idea by asking the most basic questions "who", "what", "where", "when", "how", and "why." Each question will get more and more sophisticated expanding like a spiral building a holographic structure containing all possible aspects of concepts around that seed idea.

5) An English parser to interface our architecture is necessary to fully utilize the power of natural language understanding.

6) Finally, we need to cover the most basic 300-400 concepts defining reality into our architecture. This must have a self awareness with its own personality and its own time perception (past-present-future).

VII: CONCLUSION

We believe that the relatively simple form of our algorithm for concept learning by analogy will help to better understand learning. When we introduced this architecture to other people (most of whom were outside the field), most were convinced that they could easily understand the main idea and that they were already using the approach in their daily lives. With this general architecture and a versatile set of concepts, we could possibly interface our system to robotics, to operating systems, to aircrafts, to large database systems, to search engines, and to all other possible applications. Our work could help humans to understand themselves, their mind and psychology, and their strengths and weaknesses. We can study about abstract or concrete concepts asking some basic questions. For example, what does “to love” mean? How can “to love” be explained to a computer program? Perhaps this mechanism would reflect like a mirror much of the human mind, heart, and soul.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

Allen, J., *Natural Language Understanding*, 1983.

Atkin, S., *Prolog as a Theorem Prover*, 1999.

Bernstein, T., *The Careful Writer: A Modern Guide to English Usage*, New York: Atheneum, 1977.

Carbonell, J. G., *Learning By Analogy: Formulating and Generalizing Plans From Past Experience*, Carnegie-Mellon University, 1993.

Fogiel M., *The Psychology Problem Solver*, REA, 1999.

Hofstadter, D., *The CopyCat Project: An Experiment in Nondeterministic and Creative Analogies*, Mass. MIT, 1984.

Luger, G. & Stubblefield, W., *Artificial Intelligence*, Addison-Wesley, Third Edition, 1999.

Miller, G. & Beckwith, R. & Fellbaum, C. & Gross, D. & Miller, K., *Introduction To WordNet: An On-Line Lexical Database*, 1993.

Mitchell M., *Analogy-Making as Perception*, The MIT Press, 1993.

Pinker S., *Words and Rules*, Perennial, 2000.

Schank, R.C. & Rieger, C.J., *Inference and the Computer Understanding of Natural Language,* Artificial Intelligence, *5(4)*: 373-412, 1974.

The American Heritage Dictionary, Third Edition, 1992.

Wallis, S. & Moss, S., *Efficient Forward Chaining for Declarative Rules in a Multi-Agent Modeling Language*, 1995.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. LCDR Barker
   Outgoing NPS Coordinator
   Naval Postgraduate School
   Monterey, California

4. LCDR Hunter
   Incoming POC For IW Curric
   Naval Postgraduate School
   Monterey, California

5. LCDR Stevenson
   Incoming POC For Comp-Net. Topics
   Naval Postgraduate School
   Monterey, California

6. Professor Neil Rowe
   Department of Computer Science
   Naval Postgraduate School
   Monterey, California

7. Professor John Hiles
   Department of MOVES
   Naval Postgraduate School
   Monterey, California

8. Kara Kuvvetleri Komutanligi
   Kutuphane
   Bakanliklar, Ankara, TURKEY

9. Kara Harp Okulu Komutanligi
   Kutuphane
   Dikmen, Ankara, TURKEY

10. Bilkent Universitesi Kutuphanesi
    Bilkent, Ankara, TURKEY

11. Orta Dogu Teknik Universitesi Kutuphanesi
    Balgat, Ankara, TURKEY

12. Bogazici Universitesi Kutuphanesi
    Bebek, Istanbul, TURKEY

13. Yilmaz Degirmenci
    Sarma Sok. 13/8
    Iskitler, Ankara, TURKEY